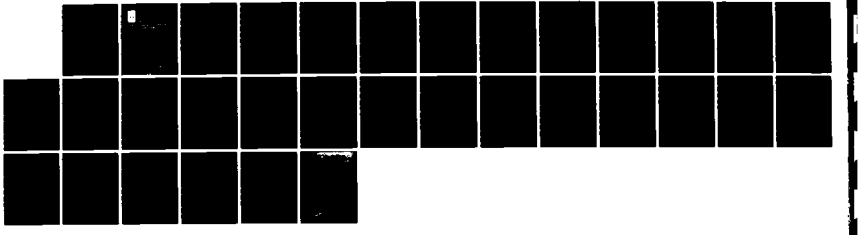
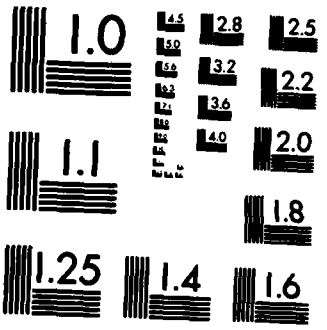
AD-A146 662 SOFTWARE CONFIGURATION MANAGEMENT ACROSS PROJECT BOUNDARIES AND IN DISTRI.. (U) ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (ENGLAND) M STANLEY 1984 1/1

UNCLASSIFIED RSRE-MEMO-3704 DRIC-BR-92718 F/G 9/2 NL





RSRE
MEMORANDUM No. 3704

ROYAL SIGNALS & RADAR ESTABLISHMENT

SOFTWARE CONFIGURATION MANAGEMENT ACROSS PROJECT BOUNDARIES AND IN DISTRIBUTED DEVELOPMENT ENVIRONMENTS

Author: Margaret Stanley

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

UNLIMITED

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 3704

TITLE: SOFTWARE CONFIGURATION MANAGEMENT ACROSS PROJECT BOUNDARIES AND IN DISTRIBUTED DEVELOPMENT ENVIRONMENTS

AUTHOR: Margaret Stanley

DATE: January 1984

SUMMARY

When software development is shared between several different sites, or host computers, or even between different development organisations, problems which are not evident in more compact development systems arise. Many of these problems manifest themselves in software configuration management. If software developed on one project is to be reused on another project then software configuration management needs to extend beyond the individual project into a wider area. This paper addresses the problems of software configuration management when sharing software between projects and between host development systems, and between projects, using an integrated Programming Support Environment. The discussion assumes the availability of some of the facilities that were proposed for inclusion in the UK CHAPSE (CHILL Ada Programming Support Environment).

UNLIMITED

TITLE: Software Configuration Management across Project Boundaries and in Distributed Development Environments.

CONTENTS

1.0 INTRODUCTION.

Software configuration management (SCM) is the name given to the task of recording and controlling all changes to the software (both code and associated documentation) throughout the life cycle of a product. In a programming support environment (PSE) with an underlying database, the software can be stored in the database and controlled using specially developed SCM tools and the facilities and checks provided by the database management system (DBMS). Reference 1 proposed a skeleton database schema and a toolset to assist in SCM for a single software project whose software development takes place on a single host PSE. This paper develops the ideas in Ref. 1 and discusses the implications both for projects that share a host PSE and for projects where the development takes place on several different host PSEs.

2.0 THE SCM DATABASE AND TOOLS.

2.1 Terminology.

When a software item is regarded as having reached a stable state, so that further changes will need to be formally agreed, it is registered in the project for which it was developed. Software items that have been registered are called software configuration items (SCIs). When a software project is planned, a number of baselines are defined, which consist of lists of the SCIs that will eventually be registered, with different baselines for different stages in the project life cycle. The SCM database is that part of the PSE database used to support software configuration management. It holds the information about project baselines; the network of SCIs planned for a baseline (called the software configuration network (SCN network)); the network of registered SCIs (the SCI network) and software change requests (SCRs).

2.2 The SCM Database.

2.2.1 Database Structure.

The SCM database for a project (see Fig. 1) will include:

1. one project entity, related to every baseline entity associated with the project;

2. baseline entities, with membership relationships to software configuration network (SCN) entities;

3. a network of SCN entities (SCN network) representing the planned SCI network, with network relationships representing the connections in the network (a membership relationship between a baseline entity and an SCN entity indicates that the SCN is a member of the baseline);

4. entities holding the body of each SCI object such as specifications, plans, source code etc.., with relationships giving the connections between them (the SCI network);

5. relationships (registration relationships) connecting the SCN network to the entities holding the body of each registered SCI, (these relationships will be created on registration of the SCI, and will bind the body of the SCI);

6. change request entities (CR entities) holding details of software change requests (SCRs), with membership relationships to the projects affected by the proposed change;

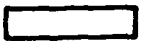
7. change relationships between CR entities and affected SCN entities and create relationships between CR entities and new SCN entities created as a result of an authorised change.

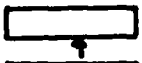
A network relationship between two SCN entities indicates a planned relationship between the registered SCIs.

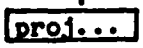
On registration of an SCI in a project a binding registration relationship will be created to precisely one SCN entity in the project . A binding relationship is one which prevents deletion of the bound object and inhibits changes to the bound object such that an attempt to change the bound object must result in a new object.

2.2.2 Symbology.

The symbols given below are used throughout this paper. New symbols will be introduced as required.

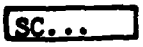
 entity.

 bound entity.



project entity.

baseline entity.



SCN entity.

source code entity.

design entity.

change request entity

- - - membership relationship (between project and baseline; or between CR entity and project; or between baseline and SCN entity).

⟶ registration relationship (between SCN and SCI, binding the SCI).

——— dependency or derivation relationship (between SCIs).

..... network relationship (between SCNs).

·—·— change relationship between CR entity and SCN to be changed.

——— create relationship between CR entity and new SCN.

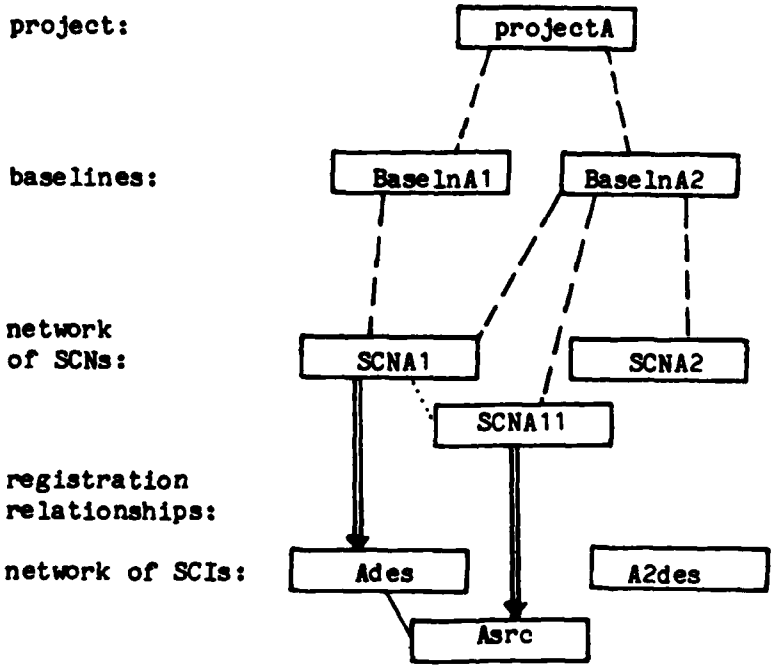


Figure 1.

In Figure 1 BaselnA2 is a baseline that includes planned configuration items SCNA1; SCNA11 and SCNA2. Ades and Asrc have been registered in projectA but A2des has not yet been registered. SCNA2 is the SCN to which it will be connected when it is registered in projectA. No SCRs (software change requests) have yet been submitted for projectA, so no CR entities are shown.

2.3 Initial And Derived Objects.

The derivation of an object is the information needed to recreate the object. Objects in the SCM database fall into two groups, initial objects and derived objects. An initial object is any object whose derivation includes some information not held in the database. For example an object created (perhaps from another database object) using an editor and information input from a terminal is an initial object. A derived object is one whose derivation is contained within the database. It is derived (created automatically) from one or more other database objects using software tools. Some attributes (called the derived attributes) of a derived entity will be the same every time the entity is derived although others, such as creation date, will be different. The only attribute values of a derived entity which may differ according to when the object is derived are those that depend on its creation date. A derived entity can always be re-created provided its derivation is retained in the database. For example, a compiled unit, created from a source code object using a compiler, and a set of compilation commands held in another database object, is a derived object.

The distinction between derived and initial objects is important because a registered initial object (except for a domain, see section on SCM and Ada) must not under any circumstances be modified or deleted from the SCM database, whereas it may, under certain circumstances, be acceptable to modify or to delete a derived object, provided its derivation is retained.

The information that an entity is used in the derivation of another entity will usually be held in the database by binding derivation relationships between the derived entity and its derivation entities. The derived entity (perhaps with the derived attributes unset) will therefore need to be retained in the SCI network to preserve the derivation information. If there is a need to save space in the SCM database it would be possible to scan the database for all derived entities and to unset the derived attributes.

2.4 SCM Tools.

The SCM toolset (see ref 1.) is a set of software tools (computer programs) operating on the PSE database to provide assistance with software configuration management. They provide facilities for creating and updating an SCM database for a project; for registration of SCIs; for searching the database to discover dependencies between SCM entities and between projects; for searching the database to identify the people responsible for given software items; for listing the content of project baselines and for progressing changes through the system.

2.5 Versions.

Since software is constantly being changed, any database object may exist in a number of different versions, and it may be that several of the versions will need to exist simultaneously. The versions that result from evolution of a database object are called successors and are an ordering of the object representations. In addition to the successor versions there will be different versions of the same thing arising, for example, because of variations in the implementation of a design, or minor differences between realisations to cater for different target hardware configurations. Versions that exist in parallel, with no ordering between them, are called variants. For example, different implementations of the body of an Ada package, both satisfying the same Ada package specification, one written to minimise core occupancy and the other to minimise execution time would be variants of the Ada package body. The successors and variants of an entity are given the same name in the database because they all represent realisations of the same thing. They are distinguished from each other by the variant name and the successor number, as shown in figure 2.

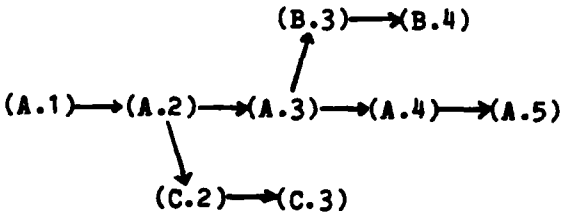


Figure 2 showing a set of successors and variants (A,B and C) of a single database object (the entity name is the same for each version and is not shown).

Successors have numeric identifiers.

Variants: B is a revision of (A.3); C is a revision of (A.2)

The entity has several "current" versions, viz. (B.4); (A.5); (C.3).

Adequate precautions will have to be taken to ensure that the correct version is used in any software configuration. It will probably be necessary to name successor and variant explicitly in any reference, rather than using the default successor and variant. The database tools may allow for a preferred variant and a preferred successor to be used as defaults if no version identification is given.

2.6 Modification Of Registered SCIs.

2.6.1 Re-registration.

Registered SCIs are bound so that authorised modification to a registered SCI will result in the creation of a new version of the SCI (except for domains, see section on SCM and Ada). Re-registration is registration of a new version of a registered SCI. It does not affect previously registered versions since the new version is a different entity. It may include additional functions such as checking that the SCN to which it is being connected is a new version of the SCN entity to which the previous version of the SCI is connected and changing an attribute of the previous SCN to indicate that the previously registered SCI has been superseded.

When a registered SCI is issued for authorised amendment, the copy will probably go through several versions before the amended SCI is ready for re-registration. It will be advisable to divorce the versioning of registered SCIs from the versioning of the issued copy to avoid unnecessary confusion. The tree structuring of versions can be used, giving the issued version a different variant name to assist in this process (see figure 3).

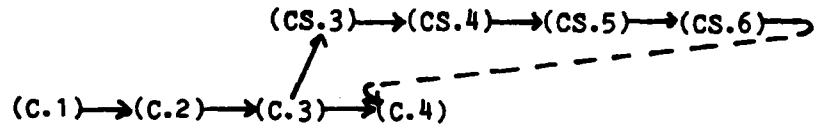


Figure 3 showing use of successors and variants when revising a registered SCI.

The SCI has two variants, C the controlled variant and CS the issued variant which is not under configuration management.

Successors have numeric identifiers.

Variants: CS is a revision of (C.3), that goes through several successors before it is ready for re-registration. (C.4) is a revision of (CS.6), created when (CS.6) is re-registered as a successor of (C.3).

When (CS.6) has been re-registered as (C.4), the successor chain for the CS variant can be deleted, since it is not under configuration management.

2.6.2 Software Change Requests (SCRs).

Consider what happens when a change is to be made to a registered software item. First a Software Change Request (SCR) (or the equivalent) must be completed, outlining and justifying the proposed changes. A change request entity (CR entity) containing the details; date and reasons for the proposed change is added to the database with a membership relationship, and approval status (initially pending) to the project entity and a change relationship to the SCN entities affected. When the SCR is submitted for approval a database search may be initiated to discover possible effects of the changes. This search will reveal any SCIs within the project that are dependent on the SCI to be changed and the names and addresses of any people who should be notified of the planned changes. Change relationships will be added to the database connecting the CR entity to other SCN entities affected by the proposed change. If the software change request is rejected the approval status is changed to rejected and no further action is required. If it is approved the approval status will be changed to approved and the affected baselines and SCN networks will be augmented by new SCN entities; by new versions of the SCN entities to be modified and by create relationships from the CR entity to every new SCN resulting from the approved change. If an approved SCR will result in modification of a registered SCI, an attribute of the SCN to be superseded will be set to indicated that a new version has been authorised and the SCI to be modified will be issued to the programmer responsible.

New SCIs resulting from approved SCRs will be submitted for registration or re-registration as appropriate. Progressing change requests will include searching the database for all create relationships connected to the CR entity and checking whether the corresponding SCIs have been registered, when the approval status associated with the relationship between the project entity and the CR entity can be changed to completed.

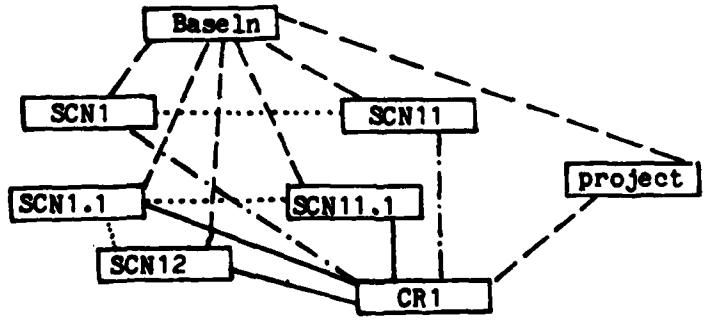


Figure 4 showing an approved CR entity and the resulting SCN network.

In figure 4 Baseln initially included SCN1 and SCN11 as members. A software change request (CR entity CR1) was raised on SCN1. The database search for dependent SCIs revealed that SCN11

would also need to be changed, so a change relationship was created from CR1 to SCN11. Implementing the approved change request involved creation of successor versions of SCN1 and SCN11 (SCN1.1 and SCN11.1) and the creation of a new entity, SCN12.

2.6.3 Example Of SCM Database Showing Registration Of A Successor.

Consider projectA, as shown in figure 1. Suppose a change request is raised in respect of Asrc. CR entity, CRA, with membership relationship to projectA (status pending) and with a change relationship to SCNA11, is added to the database (see figure 5). When the change to Asrc is authorised the status is changed to approved and the baseline involving Asrc (BaselnA2) is augmented to include a successor entity (SCNA11.1) of the SCN, SCNA11, with a create relationship to CRA. An attribute of SCNA11 is set to indicate that a successor has been approved. It is necessary to create the new SCN entity so that searches of the database can reveal whether a new version is planned that has not yet been registered.

When Asrc is issued for amendment, the variant name is changed so that successors of the issued variant can be created without concern for the successor numbers of the registered SCIs. When the new successor is re-registered, the variant name is changed back to the variant name of the original registered SCI, and the new SCI is registered as the immediate successor of the original (as Asrc.1). When Asrc.1 is registered, Asrc will remain in the SCM database but an attribute of SCNA11 will indicate that the successor has been approved and registered. The status of CRA in projectA (on the relationship between projectA and CRA) will be changed to completed because the only create relationship between CRA and the SCN network (to SCNA11.1) corresponds to a registered SCI (Asrc.1).

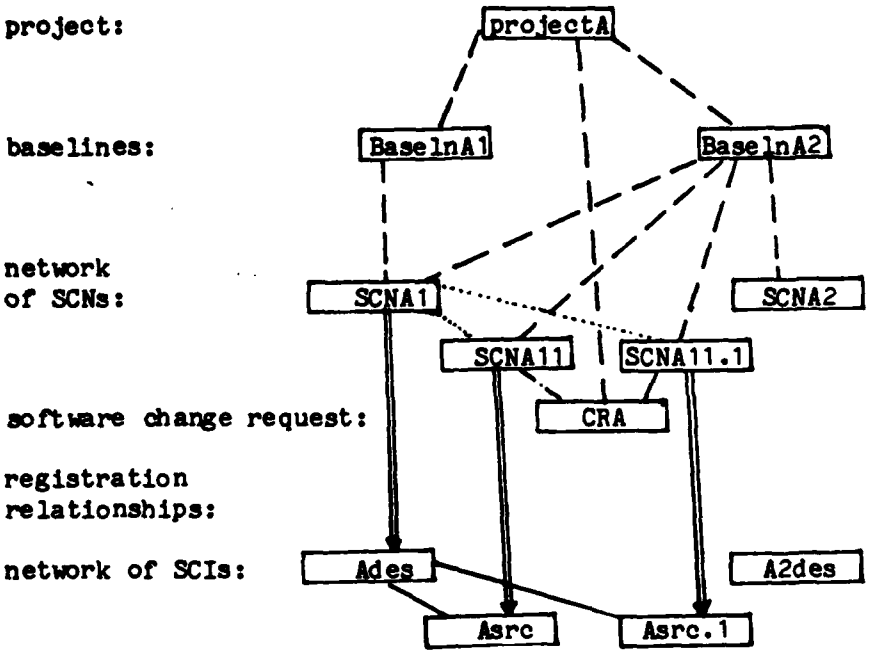


Figure 5.

3.0 SHARING A PSE BETWEEN PROJECTS.

3.1 Intersecting SCM Databases.

An organisation developing or maintaining software for a number of different projects may well wish to use the same PSE to support more than one project. Each project will build up its own SCM database (project entity; baselines; SCN network; SCI network and CR entities). The SCN networks and baselines for different projects will be disjoint even if the projects share registered software objects.

If projects do not share any SCIs then the registration of software items into the project, and the rest of configuration control will take place independently for each project. However, projects sharing a PSE database may also wish to share software. The shared software may be a software requirement, a design, some source code, some compiled code or executable programs. When a software item is registered, a registration relationship is created binding the SCI to the corresponding SCN entity of the SCN network of the project in which the SCI is being registered. Registration of an already registered SCI in another project will cause a new registration relationship to be created. It will not be affected by any previous registration (see figure 6).

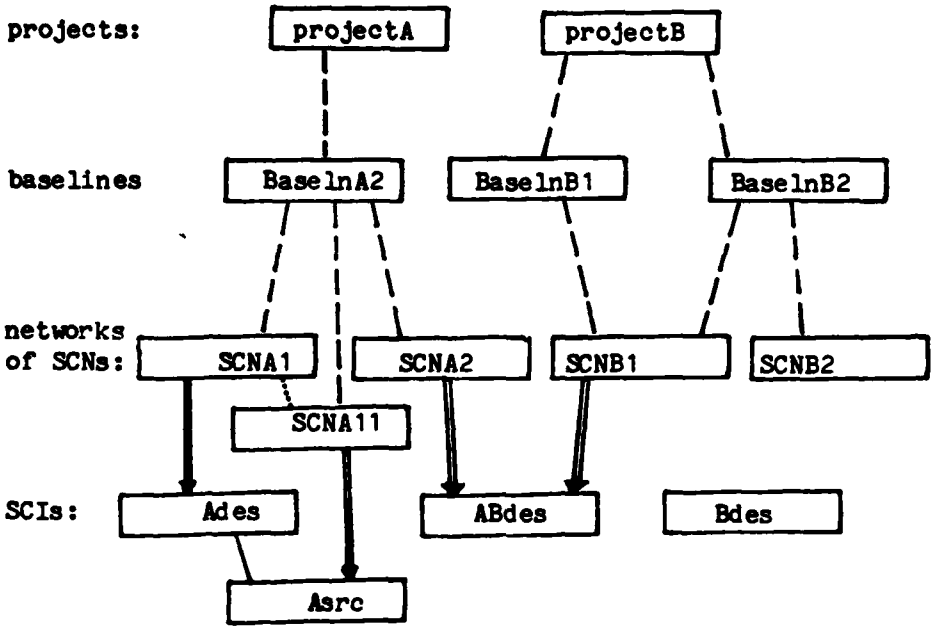


Figure 6.

Asrc is the source code to realise the design held in Ades.
ABdes is registered in both projectA and projectB.
Bdes has not yet been registered, but SCNB2 is the SCN to which it will be connected when it is registered in projectB.

3.2 Access Controls.

Access control is an essential element in configuration management. When setting up the control mechanism it may be necessary to provide the project librarian with special status giving the him greater access to registered SCIs and to SCNs and baselines than accorded to other users.

To be effective access to entities, relationships and attributes should be controlled by the DBMS. The DBMS access controls should allow access to database objects to be set individually for each entity, for each relationship and for every attribute, with access restricted according to the identity of the user requesting access, and in certain cases according to the tool being used to obtain access.

Access to objects in the SCM database that are owned by a single project will normally be limited to personnel associated with the project. Project entities, baseline entities and SCN entities are associated with precisely one project, and access to the attribute values of such entities will therefore be limited to authorised project personnel. Registered SCIs may be registered in more than one project and will therefore be accessible to personnel from each of the projects in which they are registered and CR entities affecting changes in more than one

project will need to be accessible across project boundaries.

Access controls provided by a DBMS may prevent a search of the database from locating or navigating through an object to which the user has no access right. It will not be possible to use the schema proposed in this paper for SCM in a multi-project database with shared SCI entities unless the database searches can locate entities owned by different projects, and can access the values of some attributes of some of the entities.

There are at least two ways to permit searches through several projects. One is to have a user with access permission to every object in the database who will be responsible for all searches requiring access beyond single project boundaries. The other is to ensure that the DBMS access controls permit navigation through any database object, even objects where access to individual attribute values is denied (i.e. a navigation only search). The attributes in the database that contain the names of the people responsible for projects will need to be readable across project boundaries.

3.3 Changing An SCI.

If the software is shared between projects, changes made to the shared software to suit one project should not invalidate use of the software by another project. The SCM tools appropriate to a single project PSE can easily be extended for use on a multi-project PSE, provided the access controls do not prevent access to objects owned by other projects.

The search for dependent entities needed for approval of a change request could be extended to list all other projects in which the affected SCI or any dependent SCIs are registered, and the people concerned. The SCR can then be considered by all projects in which the software is registered. If approved by any of the projects concerned, the change will be implemented in the usual way. The modified versions of the software items will be re-registered in turn in each approved project and connected to the new versions of the SCNs. If the SCR is approved by one project and rejected by another, the new version will only be re-registered for the approving project. The other project will not be updated to include the changed version.

When a successor or a new variant of a registered SCI has been registered to one project and not to another, both versions of the SCI will be registered in the database. If two versions of an SCI are current (in different projects) variants should be employed to indicate that both versions are current in some project. The tree structure of versions (successors and variants) will indicate which version was predecessor to both variants.

To assist in re-registration for all approved projects, the re-registration tool could be extended to prompt the librarians of all projects affected by a change request when a new version of the SCI is submitted for re-registration. It could also check, when a successor version is submitted for re-registration, whether the predecessor is to remain current in some project, and whether a successor for that SCI has already been registered in the database. In either case the re-registration tool could warn the user that the new version must be treated as a new variant of the SCI, in order that both versions be regarded as current in the database.

3.4 Example Of Dependency Across More Than One Project.

This example is intended to illustrated how a change in one project may affect other projects sharing the same PSE database.

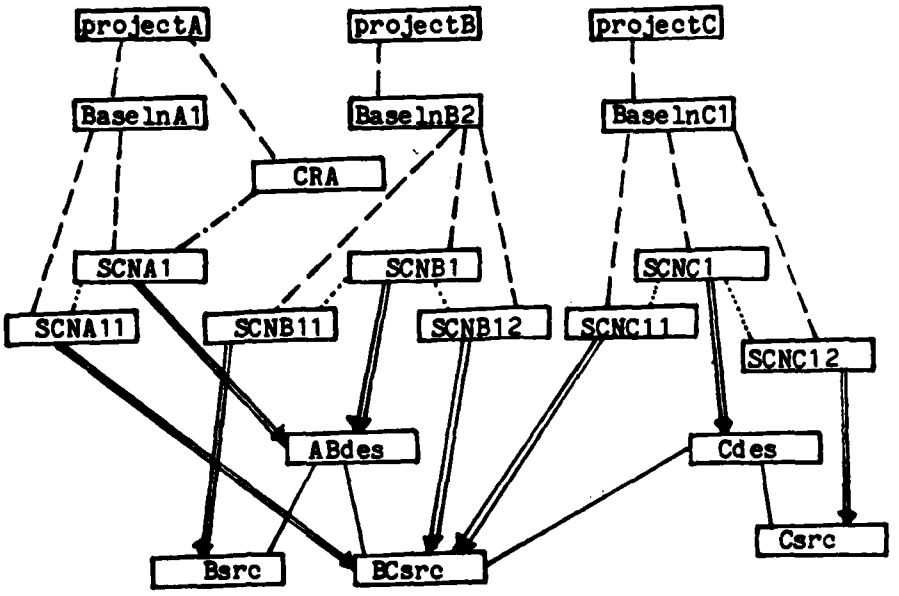


Figure 7.

In figure 7 the design ABdes is registered in both projectA and projectB. In projectB the design is realised by two source code items, Bsrc and BCsrc. The same design may have a different source code implementation (still involving BCsrc) in projectA. The different implementation is not shown in the diagram.

The design Cdes is registered in projectC and is realised in projectC by two source code items, Csrc and BCsrc. (BCsrc might be a numerical procedure needed in the implementation of ABdes in projectB and projectA and in the implementation of Cdes in projectC).

BCsrc is registered in all three projects.

Suppose a change is proposed by projectA to SCI ABdes. A change request entity (CRA), containing the details; date; and reasons for the proposed change, is added to the database with a membership relationship to projectA (status pending) and a change relationship to SCNA1. The existence of a path from SCNA1 through ABdes and then through the SCN network to baselines in two different projects (projectA and projectB) allows the database search to identify the projects directly affected by a change in ABdes. However, projectC will also be indirectly affected by a change in ABdes through the consequential change in BCsrc, so change relationships will be created from CRA to SCNA11; SCNB12 and SCNC11 (see figure 8) to indicate the potential change to all three projects as a result of a change to BCsrc. Membership relationships will be created from CRA to projectB and to projectC, each with status pending. The search for projects affected by a proposed change must cover not only the projects directly affected because they share that SCI but also any projects that share a dependent SCI. (i.e. any database search to assist in identifying affects of proposed changes in ABdes should also find projectC.)

Suppose that the software change request (CRA) is approved for projectA and for projectC but not for projectB. The status associated with the membership relationship from projectB to CRA will be changed to rejected and the status for projectA and projectC will be changed to approved. The affected baselines in projectA and in projectC will be augmented by new successors (SCNA1.2; SCNA11.2 and SCNC11.2) of the SCN entities and create relationships (not shown) will be formed from CRA to every new SCN resulting from the approved change. The SCIs to be modified (ABdes and BCsrc) will be issued to the programmers responsible and the modified SCIs will be submitted for re-registration as ABdes.2 and BCsrc.2. The re-registration tool will warn the user that ABdes and BCsrc are to remain unchanged in another project (projectB) so that a new variant (rather than a new successor) of these SCIs is required. If the new variant names are ABdes/a and BCsrc/a, the re-registration tool registers ABdes/a in projectA by linking to SCNA1.2 (see figure 8); changes the attribute of SCNA1 to indicate that the new version is now registered, and checks for any outstanding registrations associated with CRA in any project. The re-registration tool will prompt for re-registration of appropriate SCIs (i.e. BCsrc) in projectA and projectC. Since projectB is not to be changed, nothing will be re-registered for projectB. BCsrc/a will be re-registered in projectA by connecting it to SCNA11.2 and in projectC by connecting it to SCNC11.2 with appropriate changes to the SCN entities.

The fact that SCNA; SCNA11 and SCNC11 are to be superseded is shown by the existence of successors to the SCNs in the networks and the fact that the SCIs have been superseded is shown in the attributes of the superseded SCNs which are set when the changed SCIs are re-registered.

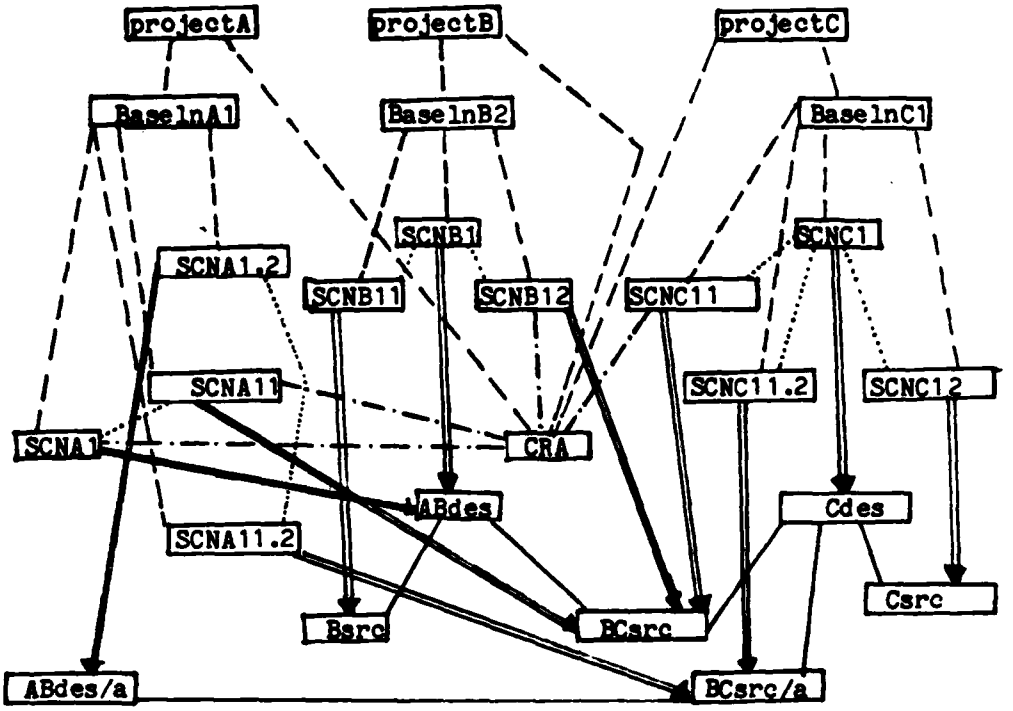
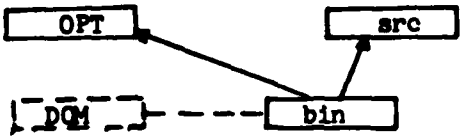


Figure 8.

3.5 SCM And Ada.

3.5.1 Domains.

If the PSE database is to be used for the development of Ada programs it will have to cater for the Ada separate compilation system (see Ref. 2), which allows separately compiled units to be combined without loss of the language related checking that would be provided were the units compiled together. In order that an Ada compiler can check that the separately compiled units are compatible, and will interface correctly, it is necessary to retain information relating to earlier compilations in a library file. The term domain has been adopted to refer to the library file of the Ada definition (ref. 2). A domain holds all the separately compiled units that will eventually be combined in an executable program, together with the associated information needed for the required checks. A domain is in fact a composite object consisting of a network of individual database entities, but it may be regarded as a single entity with a membership relationship between it and the compiled code entity of any unit compiled into it (see figure 9). (Any relationship with a domain is in fact a relationship with an entity inside the domain).



..bin.. compiled unit.

DOM composite object=domain.

- - - membership relationship (between compiled unit and domain).

——→ derivation relationship (binding derivation objects)

Figure 9.

In figure 9 the compiled unit resulting from compiling source, src, into domain DOM using compiler options held in entity OPT, is bin.
Note: there will also be relationships (not shown) between the domain and the source code entity and between the domain and the compiler options.

A domain is created and maintained by a special database tool called the domain manager, which preserves the internal structure of the domain. The domain manager is used by any tool (such as the compiler or linker) that needs to access the internal structure of a domain. The domain manager also allows a domain to be treated as a single, composite entity.

A compiled code entity may be a member of more than one domain. It is compiled into one domain and then acquired into other domains. When a compiled unit is acquired by a domain a membership relationship is created between the compiled unit and the acquiring domain. In figure 10 bin has been acquired from domain DOM into domain DOM2.

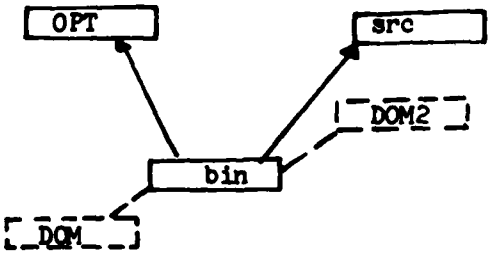


Figure 10.

When a compiled unit is removed from a domain the membership relationship between the compiled unit and the domain is severed, but the compiled unit is not necessarily deleted from the database. If all membership relationships between a compiled unit and its domains are severed, the compiled unit has no meaningful existence since it can neither be associated with other compiled units in a domain nor can it be acquired into any

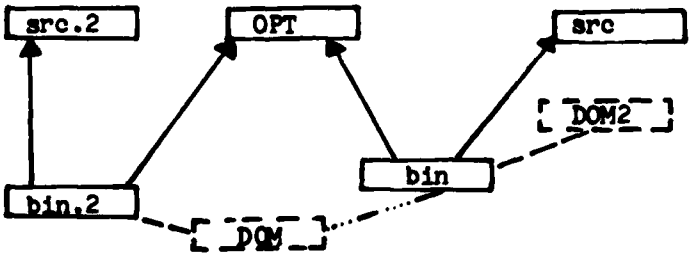
domain. It will therefore be deleted from the database by the DBMS (database management system) unless some other relationship, such as a registration relationship forces the DBMS to retain the unit. If the compiled unit is needed again it must be derived (recompiled) into a domain. Its retention in the SCM database after it has been disconnected from all the domains prevents loss of the derivation information.

The domain is different from other SCIs in that it is not in fact an entity, but is a composite object consisting of a network of entities with a membership relationship to each compiled unit in the domain. When a source entity is compiled into a domain; a compiled unit is acquired into a domain or a compiled unit is deleted from a domain, the domain changes its content and its membership relationships. An empty domain may be registered and compiled units added as they are registered. A domain is therefore permitted to change its content and relationships after registration as an SCI. No other initial objects registered in the SCM database may be modified although derived objects may be modified in that they may have the derived attribute unset, provided the derivation is retained.

A domain should not be registered in more than one project because an alteration to a domain in one project will automatically apply to any other project owning the domain, whether the alteration was authorised or not. The registration tools can ensure that a domain submitted for registration is not already registered in another project. If it is registered in another project the tool could create a new domain and acquire into it any compiled units needed in both projects.

A domain cannot hold more than one version of any unit compiled into it. If a new version of a unit is compiled into a domain the membership relationship linking the previously compiled unit to the domain is severed.

If a new version (src.2) of src (figure 10) is compiled into domain (DOM), the resulting new version (bin.2) of the compiled unit (bin) will replace bin in DOM but not in DOM2. The membership relationship between DOM and bin will be deleted (see figure 11).



.... — ..... — deleted membership relationship.

Figure 11.

If a new successor of a compiled unit is to be added to the SCM database the successor can be compiled or acquired into the domain that contains its predecessor. The predecessor will cease to be a member of the domain but its derivation will be preserved because it is registered. The historic information that it was formerly a member of the domain will be preserved in the SCN network.

If a new variant of a compiled unit is to be added to the SCM database a new domain has to be created to hold the new compiled variant so that the new variant does not disconnect the old variant from its domain (assuming each variant is current in some product). Since the new compiled variant will probably be interfaced to units already compiled into the old domain, the necessary units must be acquired from the old to the new domain.

3.5.2 Example Of Adding A New Variant To An SCM Database Involving A Domain.

This example is intended to illustrate the effect of creating a new variant and re-registering an object registered in a given baseline, where a domain is involved in the baseline.

Consider a project, projQ with a baseline BQ. BQ contains an SCN network with entities, SCQ1, SCQ2 etc.. The software objects to realise this network are two source code objects Q1src and Q3src with the source code objects compiled into domain DOM as Q1bin and Q3bin, as shown in figure 12:

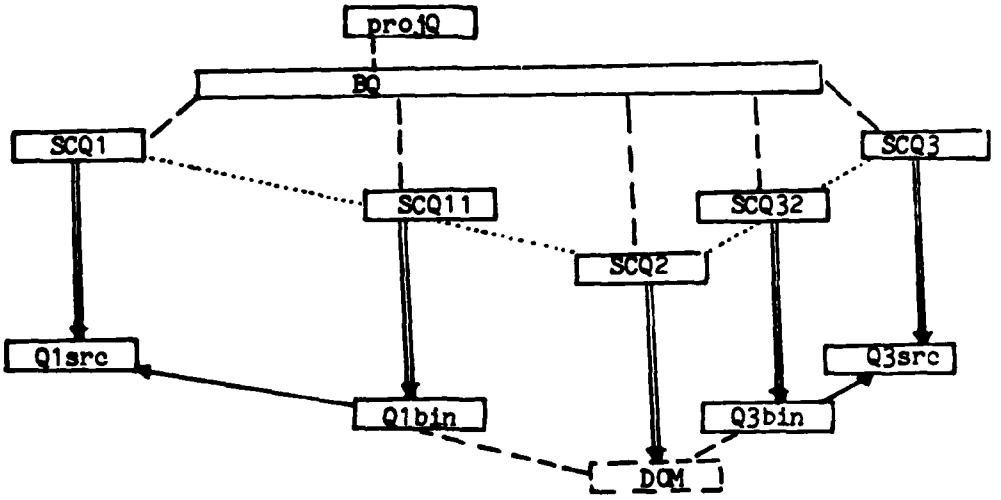


Figure 12.

Now suppose that a new variant of Q1src (viz. Q1src/a) is to be created as a result of an SCR (CR1). The baseline, BQ, will be augmented to include the new variant of SCQ1 (SCQ1/a) and its dependents (SCQ11/a). The tool to augment the baseline and to create its associated SCN network will need to form

relationships between BQ and the new variants of the SCNs and between the SCR (CR1) and the new variants of the SCNs. It will be necessary to create an SCN (SCQ2/a) for the new domain and to register a domain (DOM2) for the augmented baseline to contain the compiled units such as Q3bin, and to ensure that all registered compiled units that have no planned new variant are acquired from the parent domain (DOM). Since Q1bin is derived from (and is therefore dependent on) Q1src the new domain will also eventually contain Q1bin/a, compiled from Q1src/a. The resulting database for BQ is shown in figure 13.

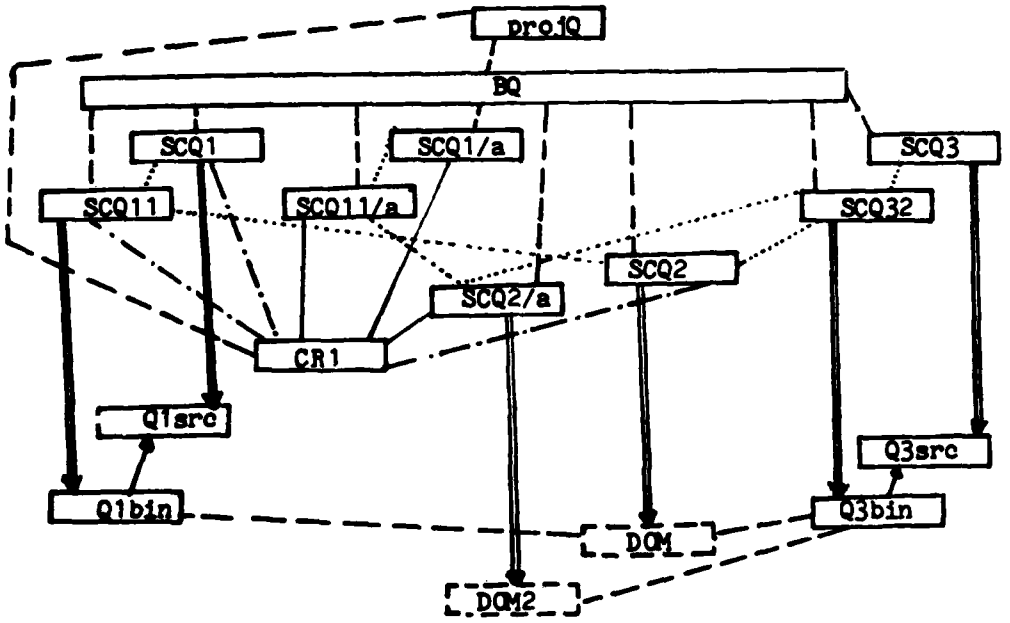


Figure 13.

When Q1src/a and Q1bin/a are ready for re-registration, the re-registration tool can acquire Q1bin/a into the new domain, DOM2, from the domain in which it was tested. The modifed software items can then be connected to the SCN network by setting up the relationships between the SCN network and the new versions (see figure 14).

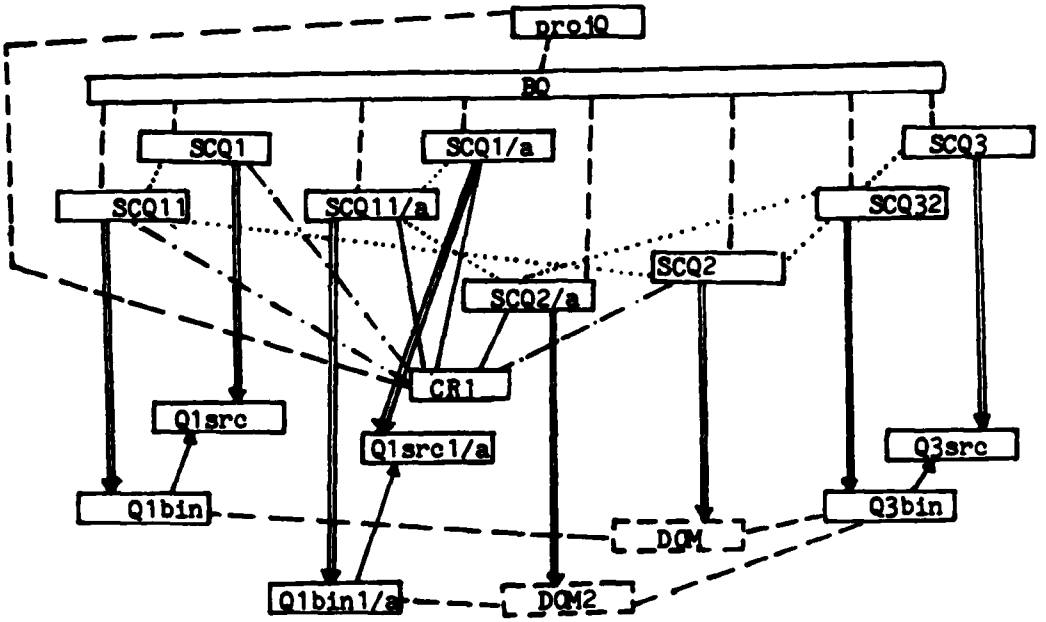


Figure 14.

Now if, for example Q1src had been registered in another project (projectR, say), then re-registration of the SCI in projQ would have no effect on the SCIs registered in projectR. If the change to a shared SCI in projQ was to result in the same change in projectR, it would be necessary explicitly to update the SCN networks in projectR in the same way as in projQ, and to re-register the affected SCIs in projectR.

3.5.3 Example Of Adding A New Successor To An SCM Database Involving A Domain.

This example is intended to illustrate the effect of creating a new successor and re-registering an object registered in a given baseline, where a domain is involved in the baseline.

Consider the SCM database shown in figure 12. Now suppose that a new successor of Q1src (viz. Q1src.1) is to be created, following approval of an SCR (CR2). The baseline, BQ, will be augmented to include the new successor of SCQ1 (SCQ1.1) and its dependents (SCQ11.1) and CR2 will have change relationships to SCQ1 and SCQ11 and create relationships to SCQ1.1 and SCQ11.1. It will not be necessary to create a new domain since Q1src.1 can be compiled into the existing domain, thus disconnecting Q1bin from the domain. The resulting database for BQ is shown in figure 15. Attributes of SCQ1 and SCQ11 will be set to indicate that the SCIs are being superseded by successors.

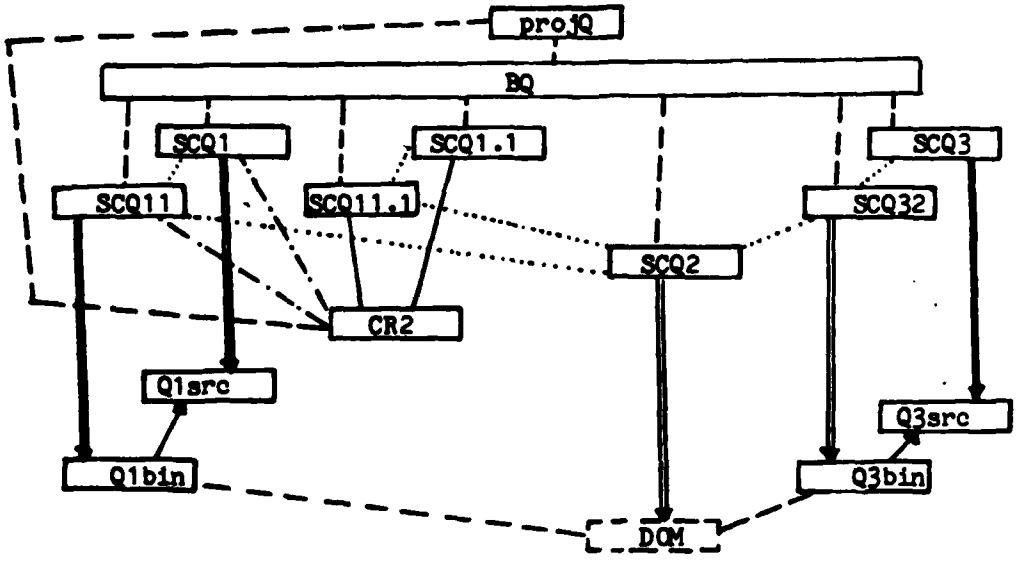


Figure 15.

When Q1src.1 and Q1bin.1 are ready for re-registration, the re-registration tool can connect the modified software items to the SCN network by setting up the registration relationships between the SCN network and the new versions (see figure 16). Q1bin will be disconnected from DQM when Q1bin.1 is added. Note that although Q1bin is not now in a domain it is retained in the SCM database by its registration relationship to SCQ11. A database search would reveal that the new successor had replaced the old because of the relationship from SCQ11 to SCQ2 which is no longer reflected in the SCI network.

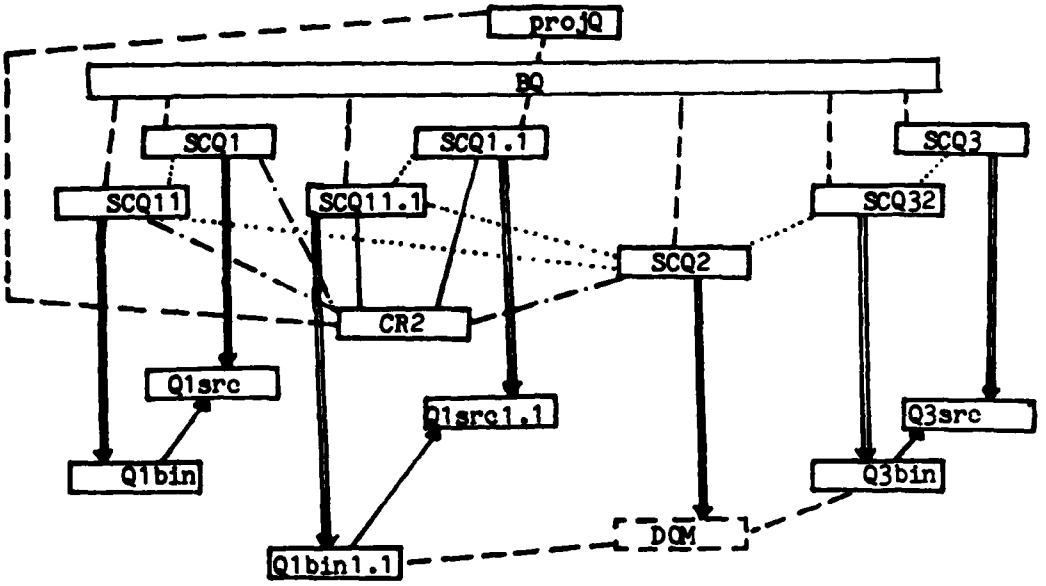


Figure 16.

Now *if*, for example Q1src had been registered in another project (projX, say),then re-registration of the SCI in projQ would have no effect on the SCIs registered in projX because DQM would not also be registered in projX.

Because Q1src and Q1bin are still current in some project (the change request was rejected for projX), the re-registration tool will only re-register the new versions as new variants (Q1src/a and Q1bin/a). The domains will still be handled as for new successors because the SCN entities (SCQ1.1 and SCQ11.1) are successors (figure 17).

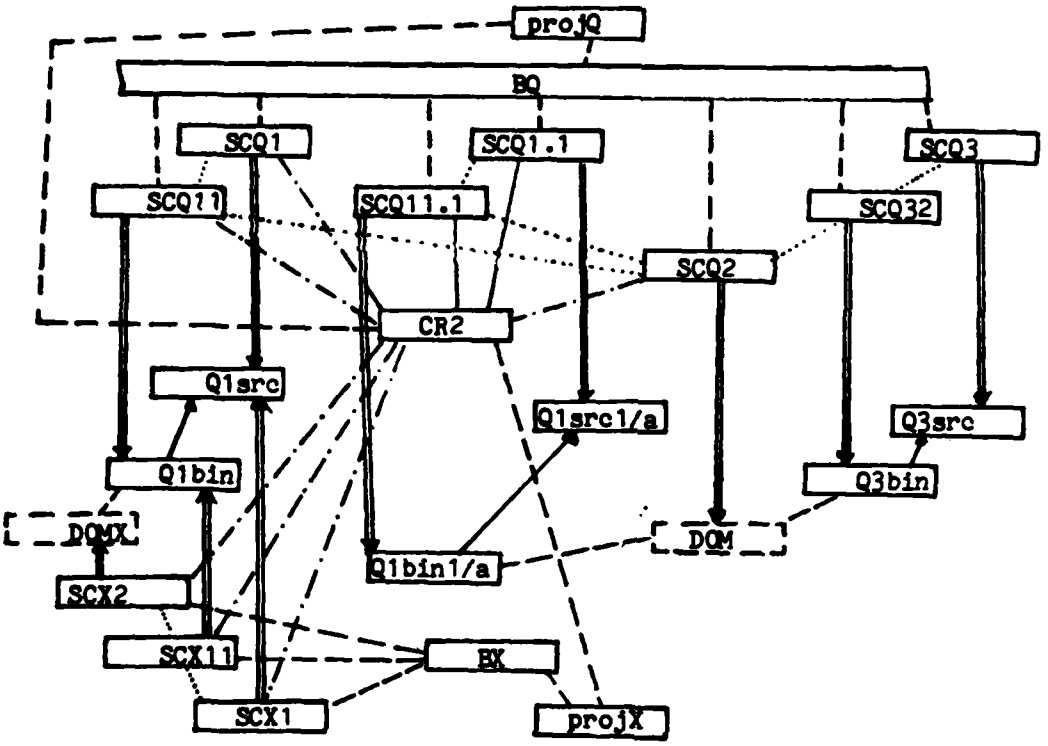


Figure 17.

Thus a change in a shared software item owned by one project cannot accidentally result in a change to the software item in another project.

3.6 Extensions To SCM Tools.

3.6.1 Access Controls And Shared SCIs.

When an SCI is registered, the registration tool may change the access permits of the SCI to restrict access to specified users using specified tools. Where an SCI is shared, access may be permitted from either project. However, the tools for issuing a copy of the SCI will need to check that all proper authorities have been obtained, including, if necessary, the authority of other projects having access rights to the SCI.

CR entities that result in changes to more than one project must be accessible from each affected project. When an SCR is found to be relevant to a project during a database search, it will be necessary to make the CR entity accessible to the project.

The access controls imposed by restricting access to specified users and specified tools must not prevent authorised access to the information that a registered SCI is shared, together with access to information to indicate by whom the SCI is shared. The information required for control of shared SCIs and of shared SCRs can only be obtained if some means is supplied of authorising access to objects owned by any project using the SCM database.

3.6.2 Baselines And Shared SCIs.

When listing the content of a baseline, there should be a facility to indicate the SCIs in the baseline that are shared with another project, when sharing started and the projects that share the SCIs.

3.6.3 Dependencies Between SCIs.

When listing SCIs dependent on any given SCI, the listing facility must include a trace of SCIs dependent through sharing of SCIs between projects and through sharing of dependent SCIs between projects.

3.6.4 Changing SCIs.

When progressing defect reports and change requests through the system the tools must include reports for all affected projects.

The facility to list the authorisations required for a given change, the SCIs affected, the people responsible and the status of the change will need extension to include authorisations from other projects.

4.0 DISTRIBUTED SOFTWARE DEVELOPMENT.

4.1 Distributed SCM.

Software for a project will usually be developed using a single host programming support environment, with a computer or a network of computers centred on a single database. The SCM database will therefore be contained in the single host database. It may, however, be necessary to spread software development over more than one host, if, for example, several companies cooperate on the development of a project or if a single company has separate development sites or because the work load is too great for a single host. If several hosts are being used, separate SCM databases should exist on each host. Even if the different hosts are electrically connected, it is not suggested that a single distributed SCM database be spread over the different hosts. The problems of distributed databases are beyond the scope of this

paper. When several SCM databases exist for a single project some method is needed of checking whether the SCM databases on the different hosts are mutually consistent. This may have to be done by visually checking the listings of the different SCM databases produced by the different hosts, although if each host is using the same PSE design and the same database schema it should be possible to develop a tool to check for mutual consistency. If inconsistency is detected it must be possible to remedy the situation by changing one of the databases.

4.2 Centralised SCM Database.

It may be desirable for overall management control to have some central form of configuration management database for the project, while still leaving each host responsible for its own configuration management. A complete SCM database might reside on one host (the central database). The individual hosts would have the master (i.e. correct) copy of their own part of the overall network, and the central database would not necessarily always be consistent with the individual host SCM databases. The central database would, however, probably be sufficiently accurate for use as a general information system but not for change control. It would, of course, be necessary to implement checks so that the differences between the central database and the individual host databases could be detected and corrected.

When any change is made to an SCM database (e.g. an SCI is registered on any host, or a baseline is amended or created) it would be necessary to have standard procedures for notifying the central database. When an SCI is registered on a separate host a dummy SCI might be registered in the central database to hold information such as the location of the actual host holding the registered SCI, the name of the registered SCI on the actual host and the date of registration.

4.3 Sharing Software Between Host PSEs.

One of the results of distributing the software development of a project or of reuse of software items by other projects or of sharing software libraries is that certain software items will need to be transfered to other hosts after registration in their local SCM database. These items will therefore appear in more that one SCM database, a supplier database and one or more customer databases. The information that such items have been exported or imported must be included in the PSE database.

4.3.1 Importing Software From An External Source.

The imported software will need to be incorporated into the PSE database and the necessary entities and relationships will need to be set up. The procedures for importing items into a PSE database will be basically similar whether the items are supplied from the same type of database on the same host, or the same type of database on a different host or from a totally foriegn environment. The tools to perform the actual transformation from one environment to another will be simpler if the two environments are similar and will differ according the the medium (e.g. electrical connections or magnetic media) used to effect the transfer. It will be useful to extend the recipient database schema to include a supplier entity type that will hold the details of the software supplier, including the host PSE; address; contact name and any relevant contractual details and restrictions on the use of the imported software. A relationship could then be set up between a supplier entity and any software supplied from an external source. The tool that transforms the imported software into a form suitable for inclusion in the PSE database will need to create an entity of appropriate type in the PSE database with a relationship to a supplier entity, and will need to set the attributes of the supplier entity if they are not already set.

4.3.2 Kinds Of Imported Software.

The software to be imported might be text such as documentation or source code or it might be compiled units or executable programs, with or without the source code from which the compiled code was originally derived.

Let us consider first the case where text objects are to be imported. Once the entity and relationship to the supplier entity have been set up, the text can be copied into an appropriate attribute of the new entity.

If the imported object is compiled code and if no source code is provided there can be no relationship to the source code, and special tools will be needed to enter such units into a PSE database. (The relationship between source text and compiled unit is not mandatory.) The problems of importing compiled code into a domain from a foriegn PSE, including making any necessary transformations to accomodate the structure of the domain and the relationships within the domain, are beyond the scope of this paper. If we assume that tools can be devised to import compiled code from a foriegn PSE, the tools could insert the compiled code into some master domain and the inserted compiled unit could be acquired from the master domain in the usual way by any project wishing to use it. The problems of importing code compiled from a language other than Ada will be similar to those involved when compiling source into a domain using the pragma "interface" (see ref. 2 para. 13.9).

If the imported object is an executable program with neither supporting compiled code objects nor source code, then the importing tool will simply need to copy it into an entity of appropriate type, transforming it if necessary to make it executable in its new environment.

4.3.3 Control Of Imported Software.

There should be no problems in including imported software items under configuration management by registering them in the usual way. The problems of importing software from external sources impinge on configuration management in that the derivation and dependency information will not be the same as if the software development was all handled from one database. A software change request that affects an imported item must be treated with special care. If the software supplier is willing and able to cooperate in making approved changes, then the tools must assist in ensuring that he is included in the list of authorities involved in approving a change. If, on the other hand, the imported software is a frozen product, any change request that implies a change to the imported software must be rejected. If such information is held in the database, the SCM tools could assist in the rejection. If a change is made in an imported software item by a supplier, the importer has the option as to whether to implement the change. This can be handled through the normal software change request procedures.

If a defect is detected in some imported software, the relationship to a supplier entity will make it easy to discover to whom the defect report should be sent.

4.3.4 Exporting Software.

If software is to be exported from a PSE,the schema should be extended to hold a customer entity type, with similar information to that held in the corresponding supplier entity type. An exported item will then have a relationship to the customer entity. Any change request processing can then ensure that lists of affected customers are available, so that, where appropriate, the customer can be included in the approval process. Similarly, when a change has been implemented, the customer can be informed, and supplied with the amended software.

Defect reports can be distributed to customers using a tool to list all affected customers.

5.0 CONCLUSIONS.

Frequently software for a project will be developed using a single host development environment and therefore configuration management will be done on that host. Even when projects share software items on the host, separate networks of baseline entities and SCNs should be maintained for each project. The sharing of the SCIs will be indicated in the database by registration relationships from the SCI to each of the SCNs that it embodies. The SCM tools will need to be extended to use these additional relationships to discover the implications of sharing SCIs when software changes are proposed or implemented. The sharing of SCIs may have indirect implications for projects that do not directly share the object to be changed, and the tools will therefore need to follow through all the dependencies of any item to be changed.

In order for the proposed SCM schema and tools to work the access controls imposed by the DBMS must permit database searches across project boundaries. This may either be achieved by permitting searches through database objects while denying access to the attributes of the objects or by giving a priveleged user access to all objects in the PSE database and requiring that all the searches for implications of proposed changes to registered software in any project be performed by the priveleged user.

If software development for a project is spread over a number of different host PSEs, separate SCM databases should exist on each host. It may also be useful to create a central database with dummy SCIs registered to represent SCIs registered on other hosts instead of importing copies of the SCIs to the central database. The central database could then be used as a source of information on the status of the project as a whole, with the individual hosts still retaining control of the detailed SCM.

It will be necessary to import and export software items to and from host PSEs. In order to control imported or exported software there should be relationships to entities representing the external host so that tools searching for possible effects of proposed changes can indicate that other host PSEs should be considered.

6.0 REFERENCES.

1. M.Stanley. "Software Configuration Management in an integrated PSE.", RSRE Memo. 3578.

2. US DoD "Reference Manual for Ada Programming Language.", July 1982.

DOCUMENT CONTROL SHEET UNLIMITED

Overall security classification of sheet ~~UNCLASSIFIED~~

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification
	Memorandum 3704		Unclassified

5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location
	ROYAL SIGNALS AND RADAR ESTABLISHMENT

5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location

7. Title SOFTWARE CONFIGURATION MANAGEMENT ACROSS PROJECT BOUNDARIES AND IN DISTRIBUTED DEVELOPMENT ENVIRONMENTS

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
STANLEY, M				

11. Contract Number	12. Period	13. Project	14. Other Reference

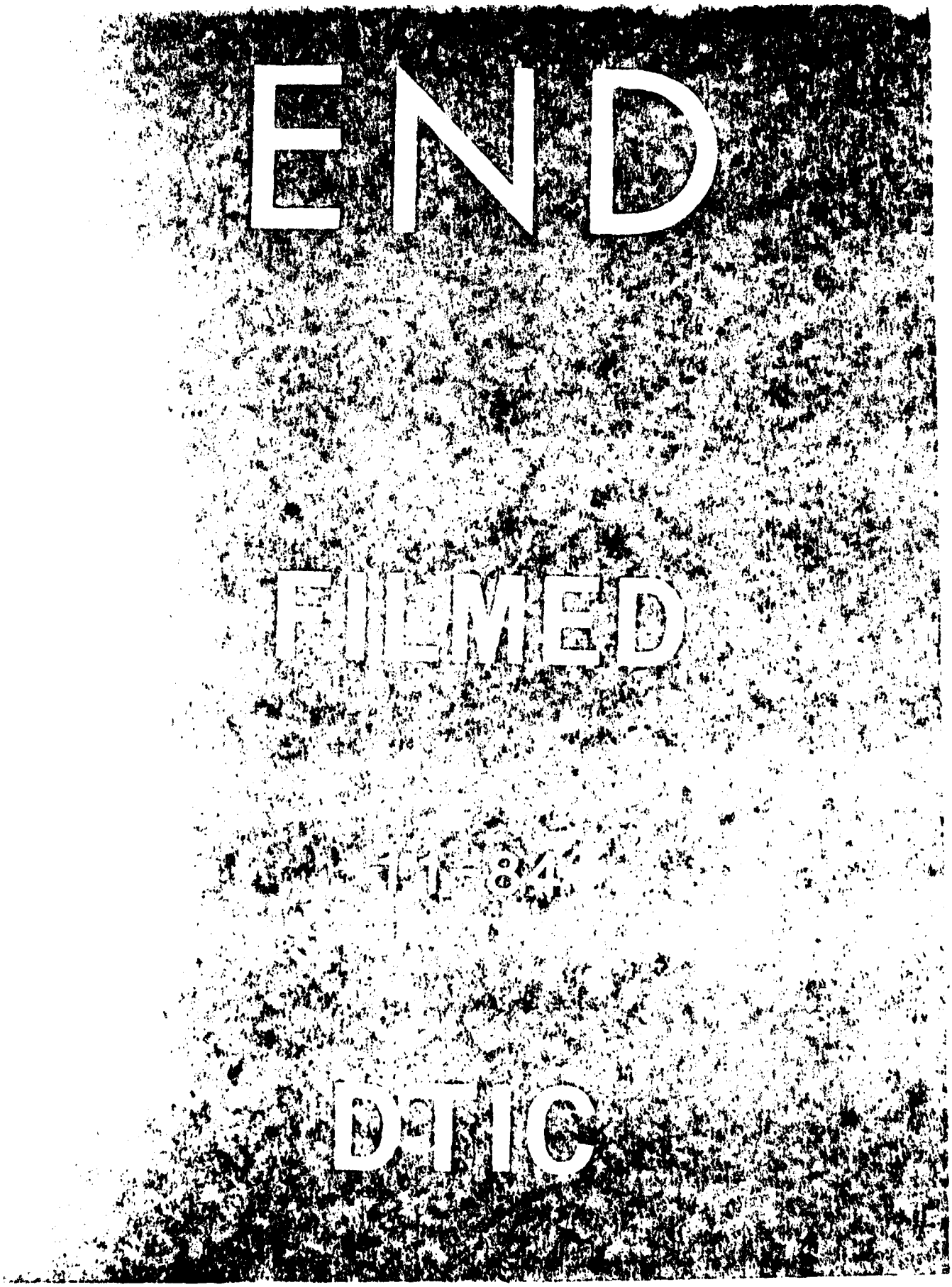
15. Distribution statement

UNLIMITED

Descriptors (or keywords)

continue on separate piece of paper

Abstract When software development is shared between several different sites, or host computers, or even between different development organisations, problems which are not evident in more compact development systems arise. Many of these problems manifest themselves in software configuration management. If software developed on one project is to be reused on another project then software configuration management needs to extend beyond the individual project into a wider area. This paper addresses the problems of software configuration management when sharing software between projects and between host development systems, and between projects, using an integrated Programming Support Environment. The discussion assumes the availability of some of the facilities that were proposed for inclusion in the UK CHAPSE (CHILL Ada Programming Support Environment).


END
FILMED
1 - 84
DTIC